How to Remove Frames from Matplotlib Plots for Cleaner Visualizations

Authored by

Mohammed looti

November 13, 2025

Decoding Matplotlib's Default Figure Structure: Frames and Spines

When employing the powerful Matplotlib library for generating scientific or analytical visualizations, the resulting graphical output invariably includes a default bounding box. This box is technically composed of four individual lines known as the **axes spines**. These spines--representing the left, right, top, and bottom boundaries--serve as the foundation for anchoring axis labels, tick marks, and grids in standard plots. While this default framing is essential for traditional data visualization tasks, it can often be superfluous or visually distracting in modern reporting contexts.

The need for frame removal typically arises when charts are embedded into professional dashboards, high-fidelity academic journal articles, or integrated web applications where a minimalist aesthetic is prioritized. In such scenarios, the explicit rectangular border provided by the default spines can clutter the figure or conflict with the surrounding design elements. Achieving a clean, focused presentation of the data itself requires precise control over these structural components, allowing the visualization to breathe within its intended environment.

Fortunately, Matplotlib offers a highly flexible and efficient method to manage the visibility of these elements. The most effective approach involves leveraging the global configuration system, specifically targeting the parameters that control the drawing of the boundary lines. By understanding how to manipulate these settings, developers and analysts can transform standard framed plots into sleek, borderless figures suitable for advanced integration.

Global Configuration via the `rcParams` Dictionary

Matplotlib's behavior and appearance are governed by a dictionary of default settings known as rcParams. This powerful mechanism allows users to set persistent configuration values for virtually every graphical property, from line thickness to font size. For the specific task of frame removal, we focus our attention on the parameters associated with the **axes spines**, which are accessed using the keys prefixed with `axes.spines`.

To globally suppress the drawing of the entire rectangular frame, we must target all four directional components individually: `axes.spines.left`, `axes.spines.right`, `axes.spines.top`, and `axes.spines.bottom`. By setting the boolean value of each of these configuration keys to **False**, we instruct Matplotlib to skip the rendering of the corresponding boundary line. This technique provides a universal solution for generating frameless charts throughout an entire Python session, eliminating the need to apply modifications to every single figure object.

The primary advantage of using the `rcParams` method is its immediacy and scope. Once these parameters are updated, all subsequently generated plots will inherit this frameless characteristic, streamlining the workflow for creating cohesive sets of visualizations. This is particularly useful when developing modular code where the plot generation logic is separated from the core

configuration setup. The following code snippet demonstrates the fundamental configuration required to disable all four axes spines prior to plotting:

import matplotlib.pyplot as plt

plt.rcParams = False
plt.rcParams = False
plt.rcParams = False
plt.rcParams = False

By executing this configuration before initiating any plot commands, we ensure that the figure starts in a borderless state. The subsequent paragraphs will illustrate this process using a reproducible example, starting with the default output to establish a clear visual comparison.

Establishing the Baseline: A Default Framed Visualization

To fully appreciate the transformation enabled by frame removal, we must first establish a reproducible graphical baseline. We will generate a simple bar chart using `matplotlib.pyplot`, often aliased as `plt`, and leverage NumPy for efficient array creation. This initial plot will be generated using Matplotlib's factory defaults, which include the standard four-sided frame surrounding the plotting area.

Our example utilizes categorical data defined by a list of letters ('A' through 'E') for the x-axis and a corresponding sequence of integers for the bar heights on the y-axis. This straightforward data structure provides a clear, unambiguous visualization that is easy to reproduce and analyze. Crucially, in this baseline demonstration, no modifications are made to the `rcParams` dictionary; the code strictly focuses on data definition and plot generation.

The code block below sets up the required libraries, defines the figure, and generates the bar plot. Notice the absence of any frame manipulation commands, allowing the default axes spines to be drawn completely:

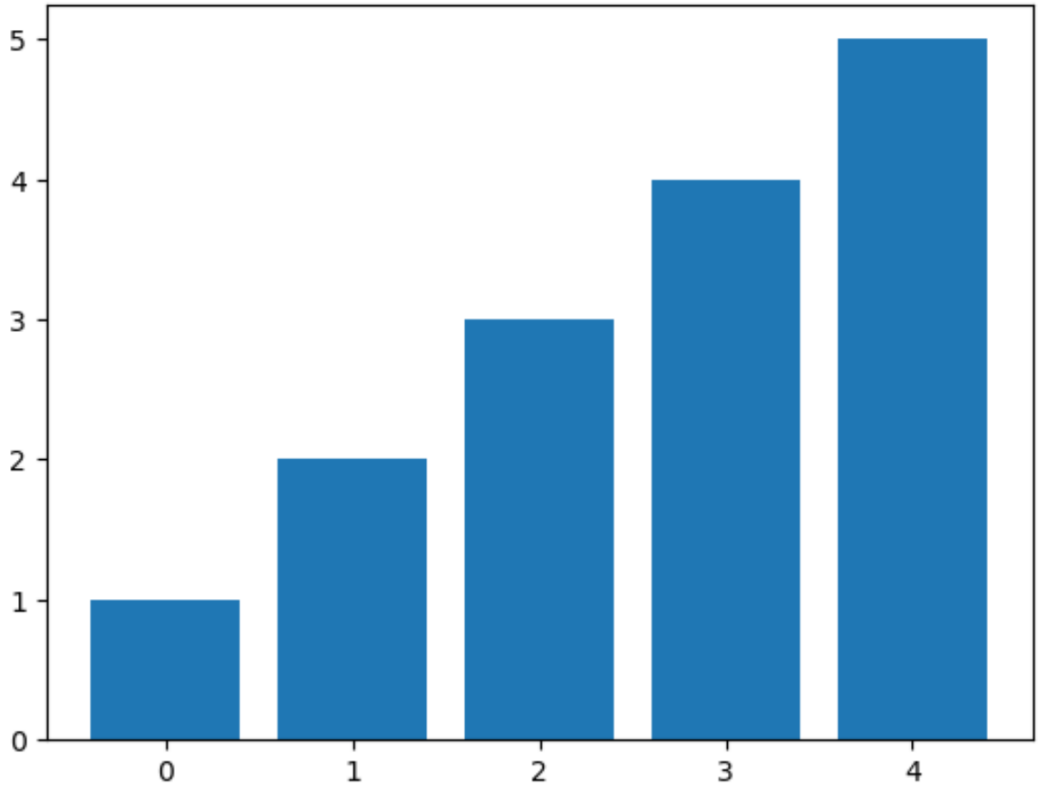
#import necessary packages
import matplotlib.pyplot as plt
import numpy as np

#initialize figure
plt.figure()

#define x and y values to be used in plot
xvals = list('ABCDE')

```
yvals = np.array(range(1, 6))

#create bar chart
position = np.arange(len(xvals))
mybars = plt.bar(position, yvals, align='center', linewidth=0)
```

Upon execution, this syntax yields the expected default output. The image below clearly displays the standard four-sided frame that encloses the data area, providing the necessary context for the axes and tick marks. This serves as the visual reference point for our subsequent modifications.



Our objective in the next step is to integrate the global configuration changes into this exact script to systematically eliminate this border, moving toward a more minimalist presentation style while preserving the integrity of the data representation.

Implementation of Total Frame Suppression for Minimalism

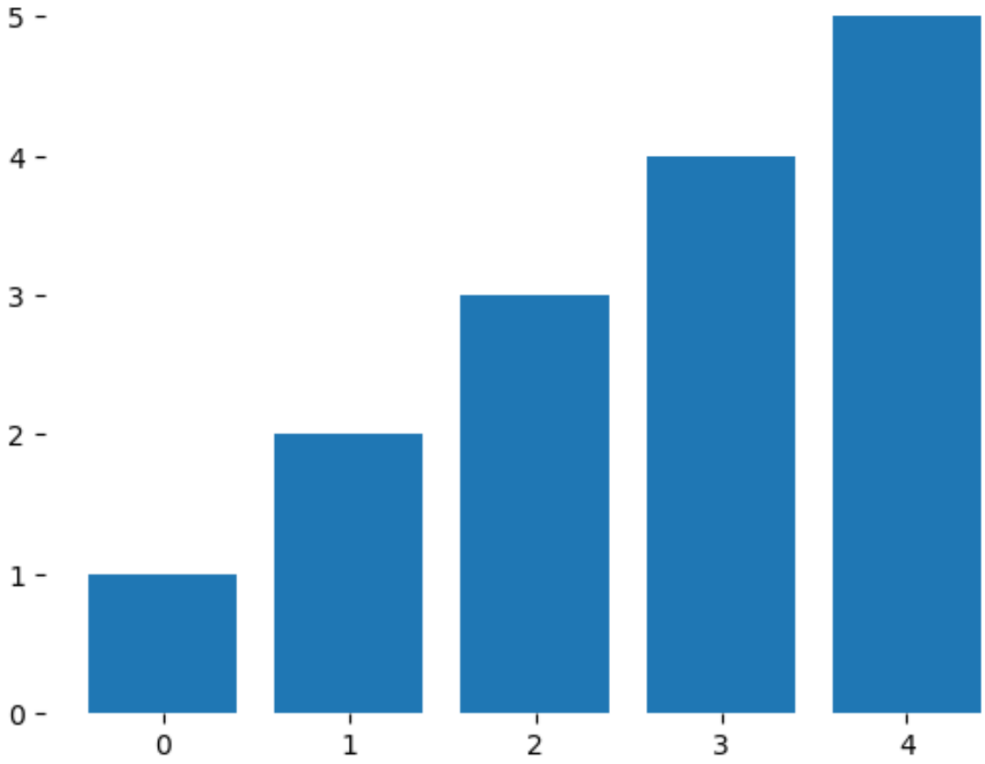
Having established the default framed appearance, we can now implement the spine removal technique using the global configuration parameters. This method involves inserting the `plt.rcParams` modifications at the beginning of the script, ensuring that the configuration is active before the axes object is fully instantiated and rendered by pyplot.

By explicitly setting `axes.spines.left`, `axes.spines.right`, `axes.spines.top`, and `axes.spines.bottom` to **False**, we instruct the visualization engine to suppress the drawing of the entire bounding box. This is particularly effective for generating graphics meant to stand alone or for dashboards where the surrounding UI elements already provide sufficient visual containment. The result is a figure that focuses solely on the plotted data, achieving a high degree of visual simplicity.

The following refined code integrates the necessary configuration lines directly before the data definition and plotting commands. Observe how these four lines immediately transform the figure environment:

```
#import necessary packages
import matplotlib.pyplot as plt
import numpy as np

#initialize figure
plt.figure()

#remove frame from each side of plot
plt.rcParams = False
plt.rcParams = False
plt.rcParams = False
plt.rcParams = False

#define x and y values to be used in plot
xvals = list('ABCDE')
yvals = np.array(range(1, 6))

#create bar chart
position = np.arange(len(xvals))
mybars = plt.bar(position, yvals, align='center', linewidth=0)
```

The resulting figure, shown below, demonstrates the successful removal of the entire enclosing border. Note that while the axes lines themselves are gone, the tick marks and labels associated with the axes remain visible. This effect creates a clean, floating visualization, though in some instances, the lack of an explicit baseline (the bottom spine) might make the connection between the tick marks and the chart data feel tenuous.



Achieving Professional Aesthetics through Selective Spine Removal

While total frame removal is often sought for extreme minimalism, the complete absence of the baseline axes can occasionally reduce the clarity of the visualization, especially when high precision or careful reading of the tick marks is required. A highly popular and professionally accepted technique in data visualization involves the selective removal of only the redundant secondary spines (top and right), thereby retaining the structural integrity provided by the primary axes (left and bottom).

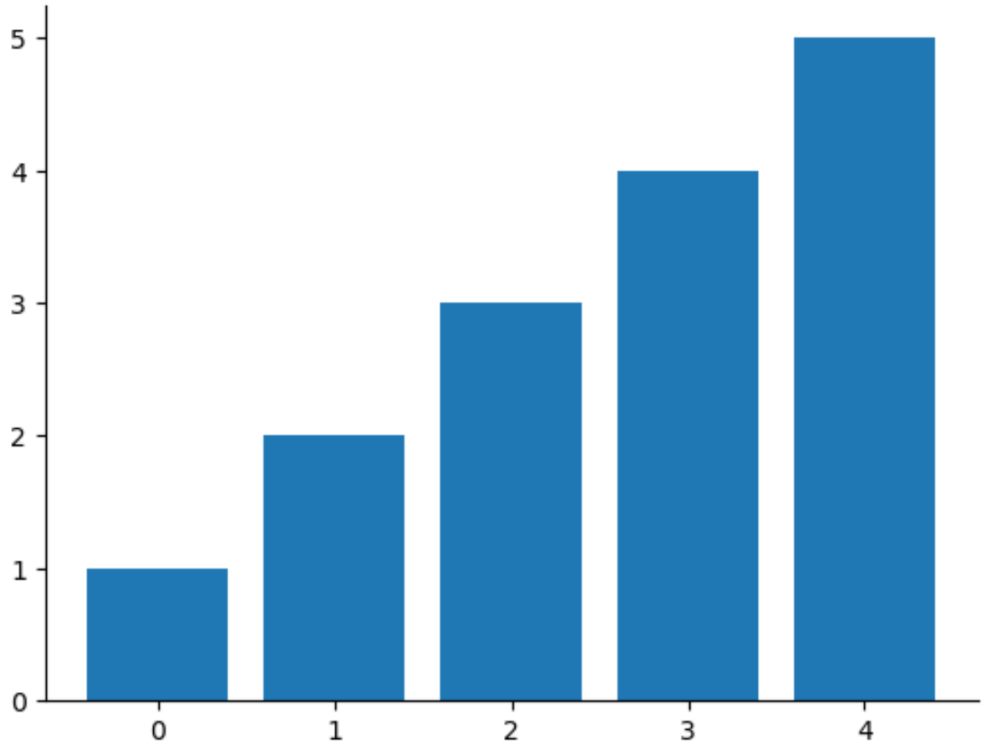
This approach maintains the essential visual anchors--the y-axis baseline for measuring height and the x-axis baseline for categorical alignment--while eliminating the unnecessary enclosure provided by the top and right lines. By preserving the left and bottom axes, we ensure that the tick marks and labels remain visually grounded and clearly associated with the plotted data.

To implement selective removal, we adjust the `plt.rcParams` configuration to ensure the visibility of the primary axes spines is set to **True**, while the visibility of the secondary axes spines is set to **False**. Specifically, we keep `axes.spines.left` and `axes.spines.bottom` active, and suppress `axes.spines.top` and `axes.spines.right`. This fine-grained control is paramount for achieving publication-quality graphics.

The revised code block below demonstrates this selective configuration:

```
#import necessary packages
import matplotlib.pyplot as plt
import numpy as np

#initialize figure
plt.figure()

#remove only right and top frames from plot
plt.rcParams = True
plt.rcParams = False
plt.rcParams = False
plt.rcParams = True

#define x and y values to be used in plot
xvals = list('ABCDE')
yvals = np.array(range(1, 6))

#create bar chart
position = np.arange(len(xvals))
mybars = plt.bar(position, yvals, align='center', linewidth=0)
```

The resulting visualization, displayed below, showcases this balanced aesthetic. The data is clearly framed by the essential axes, while the top and right boundaries are successfully removed, giving the plot a refined, open feel. Mastering this selective control over the axes spines is a key skill for advanced Matplotlib customization.



Summary of Matplotlib Frame Control Techniques

The flexibility offered by Matplotlib's configuration system provides developers with complete authority over the visual presentation of their figures. The decision to remove all spines, or selectively remove only the top and right spines, depends heavily on the context and target audience of the visualization. Both techniques utilize the powerful `plt.rcParams` global settings to achieve persistent and reproducible results.

In summary, the frame control techniques discussed offer three distinct visual outcomes:

Default Frame: All four axes spines are visible (**True**), providing traditional bounding context.
Total Suppression: All four spines are set to **False**, achieving maximum minimalism, often resulting in floating tick marks. This is best for heavily stylized charts or embedded graphics where space is limited.
Selective Removal: Only `top` and `right` spines are set to **False**, maintaining the primary x and y axes for grounding the data. This is the recommended approach for publication-quality scientific figures that require clarity and professional design.

For those seeking further mastery over their plots, exploring additional customization options within Matplotlib is highly recommended. The official documentation provides comprehensive details on manipulating other elements such as tick marks, grid lines, and labels, all of which contribute to the final appearance of a publication-ready graphic.

Additional Resources for Matplotlib Customization

Beyond managing frames, Matplotlib offers extensive capabilities for customizing every aspect of a figure. The following tutorials explore other common tasks essential for creating publication-ready graphics:

How to Remove Ticks from Matplotlib Plots

How to Change Font Sizes on a Matplotlib Plot

Featured Posts

Statistics Cheat Sheets to Get Before Your Job Interview

May 6, 2024

5 Statistical Biases to Avoid

April 25, 2024

5 Free Statistics Courses for Beginners

April 19, 2024

5 MIT Statistics Courses That Are Free

April 18, 2024

5 Free Books to Learn Statistics

April 18, 2024

How to Use the info() Method in Pandas

April 12, 2024